LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

POST TEST

Oleh:

Ramdhan Wijaya

2211102208

IF-10-M

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

1. Dosen.java

```
/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
import java.text.DecimalFormat;
abstract class dosen {
    public String nidn;
    public String nama;
    public String prodi;
    public double gaji;

    public dosen(String NIDN, String nama, String prodi, double gj) {
        this.prodi = prodi;
        this.nidn = NIDN;
        this.gaji = gj;
        this.nama = nama;

    }

    public void tampilData() {
        DecimalFormat df = new DecimalFormat("#,###.00");
        System.out.println("NIDN: " + nidn);
        System.out.println("Nama: " + nama);
        System.out.println("Prodi: " + prodi);
        System.out.println("Gaji: " + df.format(gaji));
    }

    //sebagai method abstract yang bisa di gunakan di class child childnya
    public abstract double hitungTotalGaji();
    public abstract double hitungPajak();

}
```

```
/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
import java.text.DecimalFormat;

class dosenTetap extends dosen {
    public double tunjangan;
```

```
    public dosenTetap(String NIDN, String nama, String prodi, double gj,
double tjg) {
        super(NIDN, nama, prodi, gj);
        this.tunjangan = tjg;
    }



    @Override
    public double hitungTotalGaji() {
        return (gaji - hitungPajak()) + tunjangan;
    }


    @Override
    public double hitungPajak() {
        return gaji * 0.05;
    }

    @Override
    public void tampilData() {
        DecimalFormat df = new DecimalFormat("#,###.00");
        System.out.println("====Data Dosen Tetap====");
        super.tampilData();
        System.out.println("Tunjangan: " + df.format(tunjangan));
        System.out.println("Total Gaji: " + df.format(hitungTotalGaji()));
    }
}
```

```
/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
import java.text.DecimalFormat;
class dosenTidakTetap extends dosen {
    private int jumlahTatapMuka;

    public dosenTidakTetap(String NIDN, String nama, String prodi, double gj,
int jmlTatapMuka) {
        super(NIDN, nama, prodi, gj);
        this.jumlahTatapMuka = jmlTatapMuka;
    }
```

```
    @Override
    public double hitungTotalGaji() {
        return (gaji - hitungPajak()) + (jumlahTatapMuka * 25000);
    }

    @Override
    public double hitungPajak() {
        return gaji * 0.05;
    }


    @Override
    public void tampilData() {
        DecimalFormat df = new DecimalFormat("#,###.00");
        System.out.println("====Data Dosen Tidak Tetap====");
        super.tampilData();
        System.out.println("Jumlah Tatap Muka: " + jumlahTatapMuka);
        System.out.println("Total Gaji: " + df.format(hitungTotalGaji()));
    }
}
```

```
/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
import java.text.DecimalFormat;

public class main {
    public static void main(String[] args) {
        dosen dsnTtP = new dosenTetap("06001", "Adi", "S1IF", 1000000,
100000);
        dosen dsnTdkTtp = new dosenTidakTetap("06002", "Ani", "S1IF",
500000, 8);

        dsnTtP.tampilData();
        System.out.println();
        dsnTdkTtp.tampilData();

        System.out.println("\n\n2211102208");
    }
}
```

Output :

```
run:
====Data Dosen Tetap====
NIDN: 06001
Nama: Adi
Prodi: S1IF
Gaji: 1,000,000.00
Tunjangan: 100,000.00
Total Gaji: 1,050,000.00

====Data Dosen Tidak Tetap====
NIDN: 06002
Nama: Ani
Prodi: S1IF
Gaji: 500,000.00
Jumlah Tatap Muka: 8
Total Gaji: 675,000.00


2211102208
BUILD SUCCESSFUL (total time: 0 seconds)
```

Keterangan :

Program diatas terdapat dari 4 class Dimana ada 1 class interface, 2 class child dan 1 class utama untuk run program. Pada class interface yang dilakukan adalah membuat class dosen yang mempunyai variable nidn,nama,prodi,gaji.

Selanjutnya ada class child. Jadi class child ini adalah turunan dari class dosen Dimana hubungannya class child sudah pasti class parent, sedangkan class parent belum tentu class child. Class child yang dimaksud adalah class dosenTetap.java dan dosenTidakTetap.java, class class tersebut mempunyai method dari class parentnya, yaitu tampilData(), hitungTotalGaji(),dan hitungPajak() yang bisa dipanggil Ketika class child tersebut dibuat.

Terakhir ada class main.java sebagai class utama untuk menjalankan program. Pada class utama disini kita membuat objek dsnTtP dan dsnTdkTtp dari class dosen yang dialokasikan ke memori class dosenTetap dan dosenTidakTetap yang memiliki parameter sesuai dengan class dosenTetap dan dosenTidakTetap diatas saat pembuatannya

Outputnya seperti gambar diatas mengeluarkan objek dosen yang kita buat sesuai parameter yang kita masukan. Agar tampil kita menggunakan method tampilData() dari setiap class class childnya

```
dsnTtP.tampilData();
System.out.println();
dsnTdkTtp.tampilData();
```

2. Agregasi

Processor.java

```
/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
public class Processor {
    private String name;
    private int cores;
    private double speed;

    public Processor(String name, int cores, double speed) {
        this.name = name;
        this.cores = cores;
        this.speed = speed;
    }

    public String getName() {
        return name;
    }

    public int getCores() {
        return cores;
    }

    public double getSpeed() {
        return speed;
    }

    @Override
    public String toString() {
        return "Processor{" +
                "name : '" + name + '\'' +",cores : " + cores +",speed : " + speed +'}';
    }
}
```

RAM.java

```
/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
public class RAM {
    private String type;
    private int capacity;

    public RAM(String type, int capacity) {
        this.type = type;
        this.capacity = capacity;
    }

    public String getType() {
        return type;
    }

    public int getCapacity() {
        return capacity;
    }

    @Override
    public String toString() {
        return "RAM{" +"type : '" + type + '\'' +", capacity : " + capacity +'}';
    }
}
```

Storage.java

```
/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
public class Storage {
    private String type;
    private int capacity;

    public Storage(String type, int capacity) {
        this.type = type;
        this.capacity = capacity;
    }
```

```
    public String getType() {
        return type;
    }

    public int getCapacity() {
        return capacity;
    }

    @Override
    public String toString() {
        return "Storage{" +
                "type : '" + type + '\'' +", capacity : " + capacity +'}';
    }
}
```

PC.java

```
/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
public class PC {
    private Processor processor;
    private RAM ram;
    private Storage storage;

    public PC(Processor processor, RAM ram, Storage storage) {
        this.processor = processor;
        this.ram = ram;
        this.storage = storage;
    }

    public Processor getProcessor() {
        return processor;
    }

    public RAM getRAM() {
        return ram;
    }

    public Storage getStorage() {
        return storage;
    }
```

```
    @Override
    public String toString() {
        return "PC{" +"processor= " + processor +",\nram= " + ram +",\nstorage=
" + storage +'}';
    }
}
```

Main.java

```
/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
public class Main {
    public static void main(String[] args) {
        Processor processor = new Processor("Intel Core i5-12600K", 10, 3.7);
        RAM ram = new RAM("DDR4", 16);
        Storage storage = new Storage("SSD", 512);

        PC pc = new PC(processor, ram, storage);

        System.out.println("Spesifikasi PC : ");
        System.out.println(pc);
    }
}
```

Output :

```
run:
Spesifikasi PC :
PC{processor= Processor{name : 'Intel Core i5-12600K',cores : 10,speed : 3.7},
ram= RAM{type : 'DDR4', capacity : 16},
storage= Storage{type : 'SSD', capacity : 512}}


2211102208
BUILD SUCCESSFUL (total time: 0 seconds)
```

Keterangan :

Pada program diatas adalah program yang memiliki hubungan agregasi pada class classnya. Hubungan agregasi dalam pemrograman berorientasi objek (OOP) Java menunjukkan hubungan "memiliki" antara dua kelas.

Bukti program diatas adalah program agregasi adalah Ketika salah satu atribut saya hapus maka program akan tetap bisa berjalan dengan syarat semua yang berhubungan dengan class yang dihapus pada method tampilData() juga harus dihapus contohnya seperti berikut saya akan menghapus class storage.

```
run:
Spesifikasi PC :
PC{processor= Processor{name : 'Intel Core i5-12600K',cores : 10,speed : 3.7},
ram= RAM{type : 'DDR4', capacity : 16}}


2211102208
BUILD SUCCESSFUL (total time: 0 seconds)
```

Berikut programnya :

```
/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
public class Storage {
    private String type;
    private int capacity;

    public Storage(String type, int capacity) {
        this.type = type;
        this.capacity = capacity;
    }

    public String getType() {
        return type;
    }

    public int getCapacity() {
        return capacity;
    }

    @Override
    public String toString() {
        return "Storage{" +
```

```
            "type : '" + type + '\" +", capacity : " + capacity +'}';
    }
}


public class RAM {
    private String type;
    private int capacity;

    public RAM(String type, int capacity) {
        this.type = type;
        this.capacity = capacity;
    }

    public String getType() {
        return type;
    }

    public int getCapacity() {
        return capacity;
    }

    @Override
    public String toString() {
        return "RAM{" +"type : '" + type + '\" +", capacity : " + capacity +'}';
    }
}


public class RAM {
    private String type;
    private int capacity;

    public RAM(String type, int capacity) {
        this.type = type;
        this.capacity = capacity;
    }

    public String getType() {
        return type;
    }

    public int getCapacity() {
        return capacity;
    }
```

```
    @Override
    public String toString() {
        return "RAM{" +"type : '" + type + '\" +", capacity : " + capacity +'}';
    }
}

public class Processor {
    private String name;
    private int cores;
    private double speed;

    public Processor(String name, int cores, double speed) {
        this.name = name;
        this.cores = cores;
        this.speed = speed;
    }

    public String getName() {
        return name;
    }

    public int getCores() {
        return cores;
    }

    public double getSpeed() {
        return speed;
    }

    @Override
    public String toString() {
        return "Processor{" +
                "name : '" + name + '\" +",cores : " + cores +",speed : " + speed +'}';
    }
}

public class PC {
    private Processor processor;
    private RAM ram;
    private Storage storage;

    public PC(Processor processor, RAM ram) {
        this.processor = processor;
        this.ram = ram;

    }
```

```
    public Processor getProcessor() {
        return processor;
    }

    public RAM getRAM() {
        return ram;
    }

    public Storage getStorage() {
        return storage;
    }

    @Override
    public String toString() {
        return "PC{" +"processor= " + processor +",\nram= " + ram +'}';
    }
}


public class Main {
    public static void main(String[] args) {
        Processor processor = new Processor("Intel Core i5-12600K", 10, 3.7);
        RAM ram = new RAM("DDR4", 16);
        Storage storage = new Storage("SSD", 512);

        PC pc = new PC(processor, ram);

        System.out.println("Spesifikasi PC : ");
        System.out.println(pc);

        System.out.println("\n\n2211102208");
    }
}
```

Composisi

CPU.java

```
package composisi;

/**
 *
```

```
 * @author ramadhan wijaya
 * 2211102208
 */
public class CPU {
    private String nama;
    private int kecepatan;

    public CPU(String nama, int kecepatan) {
        this.nama = nama;
        this.kecepatan = kecepatan;
    }

    public String getNama() {
        return nama;
    }

    public int getKecepatan() {
        return kecepatan;
    }
}
```

RAM.java

```
/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
public class RAM {
    private int kapasitas;

    public RAM(int kapasitas) {
        this.kapasitas = kapasitas;
    }

    public int getKapasitas() {
        return kapasitas;
    }
}
```

HP.java

```
/**
 *
```

```
 * @author ramadhan wijaya
 * 2211102208
 */
public class HP {
    private CPU cpu;
    private RAM ram;


    public HP(CPU cpu, RAM ram) {
        this.cpu = cpu;
        this.ram = ram;

    }

    public void getInfo() {
        System.out.println("Informasi HP:");
        System.out.println("CPU: " + cpu.getNama() + " (" + cpu.getKecepatan()
+ " GHz)");
        System.out.println("RAM: " + ram.getKapasitas() + " GB");
    }
}
```

Main.java

```
/**
 *
 * @author ramadhan wijaya
 * 2211102208
 */
public class main {
   public static void main(String[] args) {
        CPU cpu = new CPU("Snapdragon 8 Gen 1", 4);
        RAM ram = new RAM(12);

        HP hp = new HP(cpu, ram);
        hp.getInfo();

        System.out.println("\n\n2211102208")
   }
}
```

Output :

```
run:
Informasi HP:
CPU: Snapdragon 8 Gen 1 (4 GHz)
RAM: 12 GB

2211102208
BUILD SUCCESSFUL (total time: 0 seconds)
```

Keterangan :

Program diatas adalah program komposisi karena Ketika salah satu komponen dihapus maka program akan tidak berjalan ini yang disebut hubungan komposisi karena hubungan antar kelas adalah memiliki tapi juga wajib. Jadi apabila class CPU saya hapus maka program tidak akan berjalan karena class HP memiliki membuat class CPU didalamannya

Berikut adalah perbedaan antara komposisi dan agregasi

Fitur	Komposisi	Agregasi
Ketergantungan hidup	Kuat, tidak terpisahkan	Lemah, dapat dipisahkan
Keberadaan tanpa kelas	Tidak dapat eksis	Dapat eksis
Ketergantungan kontrol	Kelas yang memiliki mengontrol lifecycle	Kelas yang memiliki tidak kontrol penuh
Contoh	Mesin, ban, stir pada mobil	Buku, laptop, kursi di ruang belajar